Revisiting Yasuda et al.'s Biometric Authentication Protocol: Are you Private Enough?

Elena Pagnin (✉) , Jing Liu, and Aikaterini Mitrokotsa

Chalmers University of Technology, Gothenburg, Sweden,
elenap@chalmers.se

Abstract. Biometric Authentication Protocols (BAPs) have increasingly been employed to guarantee reliable access control to places and services. However, it is well-known that biometric traits contain sensitive information of individuals and if compromised could lead to serious security and privacy breaches. Yasuda et al. [23] proposed a distributed privacy-preserving BAP which Abidin et al. [1] have shown to be vulnerable to biometric template recovery attacks under the presence of a malicious computational server. In this paper, we fix the weaknesses of Yasuda et al.'s BAP and present a detailed instantiation of a distributed privacy-preserving BAP which is resilient against the attack presented in [1]. Our solution employs Backes et al.'s [4] verifiable computation scheme to limit the possible misbehaviours of a malicious computational server.

Keywords: Biometric Authentication, Verifiable Delegation, Privacy-Preserving Authentication.

1 Introduction

Biometric authentication has become increasingly popular as a fast and convenient method of authentication that does not require to remember and manage long and cumbersome passwords. However, the main advantage of biometrics, i.e., their direct and inherent link with the identity of individuals, also rises serious security and privacy concerns. Since biometric characteristics can not be changed or revoked, unauthorised leakage of this information leads to irreparable security and privacy breaches such as identity fraud and individual profiling or tracking [18]. Thus, there is an urgent need for efficient and reliable privacy-preserving biometric authentication protocols (BAPs).

The design of privacy-preserving BAPs is by itself a very delicate procedure. It becomes even more challenging when one considers the distributed setting in which a resource-constrained client outsources the computationally heavy authentication process to more powerful external entities. In this paper, we focus on Yasuda et al.'s protocol for privacy-preserving BAPs in the distributed setting [23] and show how to mitigate the privacy attacks presented by Abidin et al. [1] by employing Backes et al.'s verifiable computation scheme [4].

1.1 Background & related work

Distributed privacy-preserving BAPs usually involve the following entities: (*i*) a client/user $\mathcal{C}$, (*ii*) a database $\mathcal{DB}$, (*iii*) a computational server $\mathcal{CS}$, and (*iv*) an authentication server $\mathcal{AS}$. The granularity of roles and entities in the biometric authentication process facilitates the privacy-preservation of the sensitive information. This distributed setting, indeed guarantees that no single entity has access to both the biometric templates (fresh and stored ones) and the identity of the querying user.

Several existing proposals of privacy-preserving BAPs use the distributed setting, e.g., [5, 21–23], and make leverage on advanced cryptographic techniques such as homomorphic encryption [7, 23], oblivious transfer [8] and garbled circuits [14]. In particular, Yasuda et al.'s protocol [23] was claimed to be privacy-preserving since it is based on the distributed setting and relies on a novel somewhat homomorphic encryption scheme based on ideal lattices. Abildin et al. [1] showed that Yasuda et al.'s BAP is privacy-preserving only in the honest-but-curious model and described an algorithm that enables a malicious $\mathcal{CS}$ to recover a user's biometric template. Intuitively, Abidin et al.'s attack succeeds because $\mathcal{AS}$ does not detect that the malicious $\mathcal{CS}$ returns a value different from the one corresponding to the output of the (honest) outsourced computation, leaving space for *hill-climbing strategies* [20] that may lead to the disclosure of the stored reference biometric template.

Verifiable delegation of computation (VC) is a cryptographic primitive that enables a client to securely and efficiently offload computations to an untrusted server [11]. Verification of arbitrary complex computations was initially achieved via interactive proofs [2, 13] and then moved towards more flexible and efficient schemes such as [3, 9, 10, 19]. The setting of VC schemes is by nature distributed and thus perfectly fits the basic requirement of privacy-preserving BAPs. For this reason, Bringer et al. [6] suggested to use VC techniques to detect malicious behaviours in BAP.

In this paper, we provide the first explicit instantiation of a distributed privacy-preserving BAP which achieves security against malicious $\mathcal{CS}$ thanks to the verifiability of the delegated computation.

1.2 Our contributions

In this paper, we mitigate Abidin et al.'s attack [1] against Yasuda et al.'s privacy-preserving biometric authentication protocol [23] by the means of the verifiable computation scheme by Backes et al. [4]. We combine the two schemes in an efficient and secure way, and obtain a modification of Yasuda et al.'s protocol with strong privacy guarantees. As a result, we obtain a new BAP which builds on top of Yasuda et al.'s and is truly privacy-preserving in the distributed setting.

From a general point of view, this paper offers a strategy to transform privacy-preserving BAPs that are secure in the honest-but-curious model into schemes that can tolerate a malicious $\mathcal{CS}$ by addressing the most significant challenges in privacy-preserving BAPs: to guarantee integrity and privacy of both the data and the computation. Despite the idea of combining VC and BAP is quite natural and intuitive [6], the actual combination needs to be done carefully in order to avoid flawed approaches.

Organisation. The paper is organized as follows. Section 2 describes the background notions used in the rest of the paper. Section 3 contains our modification of Backes et al.'s VC scheme to combine it with the somewhat homomorphic encryption scheme used in [23]. Section 4 presents an improved version of Yasuda et al.'s BAP together with a security and efficiency analysis. The proposed privacy-preserving BAP incorporates the new construction of VC on encrypted data of Section 3. Section 5 is an important side-note to our contributions, as it demonstrates how naïve and straight-forward compositions of VC and homomorphic encryption may lead to leakage of private information. Section 6 concludes the paper.

2 Preliminaries

Notations. We denote by $\mathbb{Z}$ and $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$ the ring of integers and the integers modulo p, respectively. For two integers $x, d \in \mathbb{Z}$, $[x]_d$ denotes the reduction of x modulo d in the range of $[-d/2, d/2]$. We write vectors with capital letters, e.g., A, and refer to the i-th component of A as A_i. The symbol $x \xleftarrow{\$} \mathcal{X}$ denotes selecting x uniformly at random from the set $\mathcal{X}$.

We denote the Hadamard product for binary vectors as $\diamond : \mathbb{Z}_2^n \diamond \mathbb{Z}_2^n \to \mathbb{Z}_2^n$, with $A \diamond B = C$, $C_i = A_i \cdot B_i \in \mathbb{Z}_2$ for $i = 1, 2, \ldots, n$. The Hadamard product is similar to the inner product of vectors except that the output is a vector rather than an integer.

Bilinear maps. A symmetric bilinear group is a tuple $(p, \mathbb{G}, \mathbb{G}_T, g, g_T, e)$, where $\mathbb{G}$ and $\mathbb{G}_T$ are groups of prime order p. The elements $g \in \mathbb{G}$ and $g_T \in \mathbb{G}_T$ are generators of the group they belong to, and $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ is a bilinear map, i.e., $\forall A, B \in \mathbb{G}$ and $x, y \in \mathbb{Z}_p$ it holds that $e(xA, yB) = e(A, B)^{xy}$ and $e(g, g) \neq 1_{\mathbb{G}_T}$. In the setting of VC, the map e is cryptographically secure, i.e., it should be defined over groups where the discrete logarithm problem is assumed to be hard or it should be hard to find inverses. In bilinear groups there exists a natural isomorphism between $\mathbb{G}$ and $(\mathbb{Z}_p, +)$ given by $\phi_g(x) = g^x$; similarly for $\mathbb{G}_T$. Since ϕ_g and ϕ_{g_T} are isomorphisms, there exist inverses $\phi_g^{-1} : \mathbb{G} \to \mathbb{Z}_p$ and $\phi_{g_T}^{-1} : \mathbb{G}_T \to \mathbb{Z}_p$, that can be used to homomorphically evaluate any arithmetic circuit $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$, from $\mathbb{G}$ to $\mathbb{G}_T$. More precisely, there exists a map **GroupEval** (as defined in [4]):

$$\mathbf{GroupEval}(f, X_1, \dots, X_n) = \phi_{g_T}(f(\phi_g^{-1}(X_1), \dots, \phi_g^{-1}(X_n))).$$

For security, we assume ϕ_g and ϕ_{g_T} are not efficiently computable.

Homomorphic MAC authenticators. In this paper, we make use of Backes, Fiore and Reischuk's verifiable computation scheme based on homomorphic MAC authenticators [4], which we refer to as BFR. The BFR scheme targets functions f that are quadratic polynomials over a large number of variables. Figure 1 contains a succinct description of the BFR scheme.

KeyGen$(\lambda) \to (ek, vk)$**:** Given the security parameter λ, the key generation algorithm outputs a secret verification key vk and a public evaluation key ek.

Auth$(vk, L, m) \to \sigma$: Given the secret verification key vk, a multi-label $L = (\Delta, \tau)$ and a valid message m, this algorithm outputs an authentication tag σ.

Ver$(vk, \mathcal{P}_\Delta, m, \sigma) \to \{0, 1\}$: Given the secret key vk, a multi-label program $\mathcal{P} = ((f, \tau_1, \dots, \tau_n), \Delta)$, a valid message m and a tag σ, the verification algorithm returns an acceptance bit acc: "0" for rejection and "1" for acceptance.

Eval$(ek, f, \boldsymbol{\sigma}) \to \sigma$**:** Given the public evaluation key ek, a circuit of a quadratic polynomial f and a vector of tags $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_n)$, the evaluation algorithm produces a new tag $\sigma \leftarrow \mathbf{GroupEval}(f, \sigma_1, \dots, \sigma_n)$. The evaluation of **GroupEval** proceeds gate-by-gate through the arithmetic circuit of f with the following rules:

Fan-in-2 addition gate:

i. $X_1, X_2 \in \mathbb{G}$, output $X = X_1 \cdot X_2 = g^{x_1} \cdot g^{x_2} = g^{x_1+x_2} \in \mathbb{G}$.

ii. $\hat{X}_1, \hat{X}_2 \in \mathbb{G}_T$, output $\hat{X} = \hat{X}_1 \cdot \hat{X}_2 = e(g, g)^{x_1} \cdot e(g, g)^{x_2} = e(g, g)^{x_1+x_2} \in \mathbb{G}_T$.

iii. $\hat{X}_1 \in \mathbb{G}_T$, $X_2 \in \mathbb{G}$, output $\hat{X} = \hat{X}_1 \cdot e(X_2, g) = e(g, g)^{x_1+x_2} \in \mathbb{G}_T$.

iv. $X_1 \in \mathbb{G}$, $\hat{X}_2 \in \mathbb{G}_T$, output $\hat{X} = e(X_1, g) \cdot \hat{X}_2 = e(g, g)^{x_1+x_2} \in \mathbb{G}_T$.

Fan-in-2 multiplication gate:

i. $X_1, X_2 \in \mathbb{G}$, output $\hat{X} = e(X_1, X_2) = e(g, g)^{x_1 x_2} \in \mathbb{G}_T$.

ii. $X_1 \in \mathbb{G} \cup \mathbb{G}_T$, $c \in \mathbb{Z}_p$ constant, output $X = (X_1)^c = e(g, g)^{x_1 c} \in \mathbb{G}_T$.

The output of **GroupEval** is the output of the last gate of the arithmetic circuit.

Fig. 1: The BFR verifiable delegation of computation scheme.

For further details we refer the reader to the main paper [4].

Homomorphic encryption. Let $\mathcal{M}$ denote the space of plaintexts that support an operation $\boxdot$, and $\mathcal{C}$ be the space of ciphertexts with $\odot$ as operation. An encryption scheme is said to be homomorphic if for any key, the encryption function **Enc** satisfies: $\mathbf{Enc}(m_1 \boxdot m_2) \leftarrow \mathbf{Enc}(m_1) \odot \mathbf{Enc}(m_2)$, for all $m_1, m_2 \in \mathcal{M}$, where $\leftarrow$ means computed without decryption. In this paper, we only use Somewhat Homomorphic Encryption schemes (SHE). As the name suggests these schemes only support a limited number of homomorphic operations, e.g., indefinite number of homomorphic additions and finite number of multiplications. The choice to use SHE instead of Fully Homomorphic Encryption [12] is due to efficiency: SHE, if used appropriately, can be much faster and more compact [15].

The Yasuda et al. protocol. Yasuda et al. [23] proposed a privacy-preserving biometric authentication protocol that targets one-to-one authentication and relies on somewhat homomorphic encryption based on ideal lattices. Two packing methods facilitate efficient calculations of the secure Hamming distance, which is a common metric used for comparing biometric templates. The protocol uses a distributed setting with three parties: a client $\mathcal{C}$, a computation server $\mathcal{CS}$ (which contains the database $\mathcal{DB}$) and an authentication server $\mathcal{AS}$. The protocol is divided into three phrases.

Setup Phase: $\mathcal{AS}$ generates the public key pk and the secret key sk of the SHE scheme in [23]. $\mathcal{AS}$ gives pk to $\mathcal{C}$ and $\mathcal{CS}$ and keeps sk.

Enrollment phase: $\mathcal{C}$ provides a feature vector A from the client's biometric data (e.g., fingerprints), runs the type-1 packing method and outputs the encrypted feature vector $\mathsf{vEnc}_1(A)$. The computation server stores $(\mathsf{ID}, \mathsf{vEnc}_1(A))$ in $\mathcal{DB}$ as the reference template for the client ID.

Authentication phase: upon an authentication request, $\mathcal{C}$ provides a fresh biometric feature vector B encrypted with the type-2 packing method and sends $(\mathsf{ID}, \mathsf{vEnc}_2(B))$ to the computational server. $\mathcal{CS}$ extracts from the database the tuple $(\mathsf{ID}, \mathsf{vEnc}_1(A))$ using ID as the search key. $\mathcal{CS}$ calculates the encrypted Hamming distance ct_{HD} and sends it to the authentication server. $\mathcal{CS}$ decrypts ct_{HD} and retrieves the actual Hamming distance $\mathsf{HD}(A, B) = \mathsf{Dec}(sk, ct_{\mathsf{HD}})$. $\mathcal{AS}$ returns yes if $\mathsf{HD}(A, B) \leq \kappa$ or no if $HD(A, B) > \kappa$, where κ is the predefined accuracy threshold of the authentication system.

Figure 2 depicts the authentication phase of Yasuda et al.'s BAP. For additional details on biometric authentication protocols and systems we refer the reader to [16].

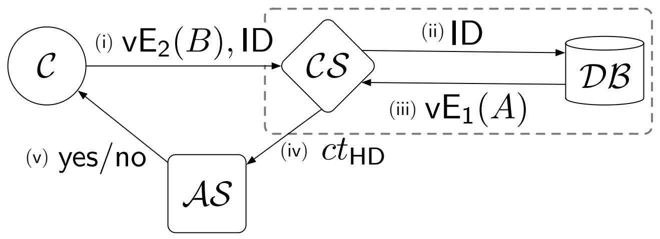

Fig. 2: Authentication phase in the Yasuda et al.'s BAP [23].

3 Combining the BFR and the SHE schemes

In this section, we describe how to efficiently combine the verifiable computation scheme BFR by Backes et al. [4] with the somewhat homomorphic scheme SHE by Yasuda et al. [23]. We call the resulting scheme BFR+SHE. Our motivation for defining this new scheme is to build a tailored version of BFR that we insert in Yasuda et al.'s biometric authentication protocol to mitigate the template recovery attack of [1].

As a preliminary step, we explain the most challenging point of the combination of the two schemes: the ring range problem. This problem rises because the elements and operations in BFR and SHE are defined over two different rings. This passage is quite mathematical, but it is necessary to guarantee the correctness of our composition BFR+SHE, presented later on in this section.

3.1 The ring range problem

The most significant challenge in combining the Backes et al. VC scheme with the Yasuda et al. SHE scheme is the different range of the base rings. While BFR handles all operations in $\mathbb{Z}_p$, where p is a prime, the operations in the SHE scheme [23] are handled in $\mathbb{Z}_d$, where d is the resultant of two polynomials. Therefore, in our BFR+SHE scheme, we need to tweak the input data if there is a mismatch in the ranges. In the calculation of the secure Hamming distance, there is a constant term equal to -2, which lives in $[-d/2, d/2)$ but not in $[0, p)$. In order to verify and generate proper tags, we can write -2 as $D = (d-2) \mod p$. Furthermore, we need to check the impact of the range difference to the verification carried out by the client. The first equation in the **Ver** algorithm of BFR is:

$$ct_{\mathsf{HD}} = y_0^{(\mathsf{HD})}, \tag{1}$$

where $ct_{\mathsf{HD}} \in \mathbb{Z}_d$ and $y_0^{(HD)} \in \mathbb{Z}_p$. In our instantiation, the term ct_{HD} corresponds to the encrypted Hamming distance between the fresh and the reference templates, while $y_0^{(\mathsf{HD})}$ is a component of the final authentication tag. As long as $d \neq p$, Equation (1) is not satisfied even when the computation is carried out correctly.

We present a general solution to this problem. For simplicity, we assume $p < d$ (as the tag size should be ideally small), although the reasoning also applies when $p > d$ by swapping the place of p and d. Our solution relies on keeping track of the dividend. Given a stored template $\alpha \in \mathbb{Z}_d$ and a fresh template $\beta \in \mathbb{Z}_d$, both encrypted, we have that: $\alpha = \alpha' + mp, \alpha' = \alpha \mod p \in \mathbb{Z}_p$, $\beta = \beta' + kp$ and $\beta' = \beta \mod p \in \mathbb{Z}_p$.

Let $\mathsf{SWHD}(x, y)$ be the arithmetic circuit for calculating the encrypted Hamming distance without the final modulo d. Let $c = \mathsf{SWHD}(\alpha, \beta)$ and $c' = \mathsf{SWHD}(\alpha', \beta')$, we can derive: $\mathsf{SWHD}(\alpha, \beta) \mod p = \mathsf{SWHD}(\alpha', \beta') \mod p$; and $c = \ell \cdot p + c'$. The value ℓ is the dividend. In our case of study, we want to perform the comparison between the Hamming distance (of the biometric templates) and the threshold κ which determines if the templates match, i.e., the client is authenticated, or not. To this end, we would track $\ell \mod d$ instead of ℓ directly. The reason is that ℓ contains more information and would lead to a privacy leak. Relating back to equation (1) we have: $ct_{\mathsf{HD}} = c \mod d \in \mathbb{Z}_q$ and $y_0^{(\mathsf{HD})} = c' \in \mathbb{Z}_p$. Given $c = \ell \cdot p + c'$, it holds that:

$$\begin{aligned} ct_{\mathsf{HD}} &= c \mod d = (\ell * p + c') \mod d \\ &= c' \mod d + (\ell \mod d) \cdot (p \mod d) \\ &= (y_0^{(\mathsf{HD})} \mod d) + (\ell \mod d) \cdot (p \mod d) \end{aligned} \tag{2}$$

Thus, if we define $\ell_d := (\ell \mod d) \cdot (p \mod d)$, the verification equation in (1) becomes $ct_{\mathsf{HD}} = y_0^{(\mathsf{HD})}(\mod d + \ell_d)$, which is satisfied whenever ct_{HD} is computed correctly (as we show in Section 3.3).

3.2 Our $\mathsf{BFR+SHE}$ scheme

To facilitate the intuition of how we incorporate $\mathsf{BFR+SHE}$ in Yasuda et al.'s BAP we describe the algorithms of $\mathsf{BFR+SHE}$ directly in the case the encrypted vectors are biometric templates:

$\mathsf{BFR+SHE}$.KeyGen(λ): The key generation algorithm of $\mathsf{BFR+SHE}$ runs $\mathsf{SHE}.\mathbf{KeyGen}(\lambda) \to (\mathsf{pk}, \mathsf{sk})$ and $\mathsf{BFR}.\mathbf{KeyGen}(\lambda) \to (ek, vk)$. The output is the four-tuple (ek, pk, sk, vk).

$\mathsf{BFR{+}SHE}.\mathbf{Enc}(pk, A, phase)$**:** The encryption algorithm takes as input the (encryption) public key pk, a plaintext biometric template $A \in \{0,1\}^{2048}$ and a $phase \in \{1,2\}$ to select the appropriate packing method. It outputs the ciphertext ct computed as $ct = \mathsf{vEnc}_{phase}(A)$, using the type-*phase* packing method of the SHE scheme.

$\mathsf{BFR{+}SHE}.\mathbf{Auth}(vk, L, ct)$**:** on input the verification key vk, a ciphertext ct and a multi-label $L = (\Delta, \tau)$, with Δ the *data set identifier* (e.g., the client's ID) and τ the *input identifier* (e.g., "stored biometric template" or "fresh biometric template"); this algorithm outputs $\sigma \leftarrow \mathsf{BFR}.\mathbf{Auth}(vk, L, ct)$, with $\sigma = (y_0, Y_i, 1) = (ct, F_K(\Delta, \tau) \cdot g^{-ct})^{1/\theta})$, where the value θ and the function F_K are defined in vk.

$\mathsf{BFR{+}SHE}.\mathbf{Comp}(pk, ct_1, ct_2)$**:** The compute algorithm takes as input the encryption public key pk, and two ciphertexts ct_1, ct_2, which intuitively correspond to the encryptions $\mathsf{vEnc}_1(A)$ and $\mathsf{vEnc}_2(B)$ respectively. The output is the encrypted Hamming distance HD calculated as: $ct_{\mathsf{HD}} = C_2 \cdot \mathsf{vEnc}_1(A) + C_1 \cdot \mathsf{vEnc}_2(B) + (-2 \cdot \mathsf{vEnc}_1(A) \cdot \mathsf{vEnc}_2(B)) \in \mathbb{Z}_d$, where $C_1 := \left[\sum_{i=0}^{n-1} r^i\right]_d$ and $C_2 := [-C_1 + 2]_d$ and r, d are extracted from pk. To solve the ring range problem described in Section 3.1 we compute ℓ_d as follows. Let c be the result of the (encrypted) Hamming distance computation without the final modulo d. Then $c' = c \mod p$ and $c = \ell p + c'$, where c' is a component in the authentication tag and ℓ is a dividend. We compute $\ell_d = \ell \mod d = (c - [c \mod p])/p \mod d$. The output is $(ct_{\mathsf{HD}}, \ell_d)$

$\mathsf{BFR{+}SHE}.\mathbf{Eval}(ek, pk, \sigma_1, \sigma_2)$**:** The evaluation algorithm takes as input the evaluation key ek, the ecryption public key pk, and two tags, which intuitively correspond to the authenticators for the two biometric templates, A, B. In our case of study, the function to be evaluated is fixed to be $f = \mathsf{HD}$ the Hamming distance. This algorithm outputs $\sigma_{\mathsf{HD}} = (y_0, Y_1, \hat{Y}_2) \leftarrow \mathsf{BFR}.\mathbf{Eval}(ek, \mathsf{HD}, (\sigma_1, \sigma_2))$.

In details, every input gate accepts either two tags $\sigma_A, \sigma_B \in (\mathbb{Z}_p \times \mathbb{G} \times \mathbb{G}_T)^2$, or one tag and a constant $\sigma, c \in ((\mathbb{Z}_p \times \mathbb{G} \times \mathbb{G}_T) \times \mathbb{Z}_p)$. The output of a gate is a new tag $\sigma' \in (\mathbb{Z}_p \times \mathbb{G} \times \mathbb{G}_T)$, which will be fed into the next gate in the circuit as one of the two inputs. The operation stops when the final gate of f is reached and the resulting tag σ_{HD} is returned. A tag has the format $\sigma^{(i)} = (y_0^{(i)}, Y_1^{(i)}, \hat{Y}_2^{(i)}) \in \mathbb{Z}_p \times \mathbb{G} \times \mathbb{G}_T$ for $i = 1, 2$ (indicating the two input tags), which corresponds respectively to the coefficients of (x^0, x^1, x^2) in a polynomial. If $\hat{Y}_2^{(i)}$ is not defined, it is assumed that it has value $1 \in \mathbb{G}_T$. Next we define the specific operations for different types of gates:

- **Addition.** The output tag $\sigma' = (y_0, Y_1, \hat{Y}_2)$ is calculated as:
$$y_0 = y_0^{(1)} + y_0^{(2)}, \quad Y_1 = Y_1^{(1)} \cdot Y_1^{(2)}, \quad \hat{Y}_2 = \hat{Y}_2^{(1)} \cdot \hat{Y}_2^{(2)}.$$
- **Multiplication.** The output tag $\sigma' = (y_0, Y_1, \hat{Y}_2)$ is calculated as:
$$y_0 = y_0^{(1)} \cdot y_0^{(2)}, \quad Y_1 = Y_1^{(1)} \cdot Y_1^{(2)}, \quad \hat{Y}_2 = e(\hat{Y}_1^{(1)}, \hat{Y}_1^{(2)}).$$
Since the circuit f has maximum degree 2, the input tags to a multiplication gate can only have maximum degree 1 each.
- **Multiplication with constant.** The two inputs are one tag σ and one constant $c \in \mathbb{Z}_p$. The output tag $\sigma' = (y_0, Y_1, \hat{Y}_2)$ is calculated as: $y_0 = c \cdot y_0^{(1)}, \;\; Y_1 = (Y_1^{(1)})^c, \;\; \hat{Y}_2 = (\hat{Y}_2^{(1)})^c$.

$\mathsf{BFR{+}SHE}.\mathbf{Ver}(vk, \mathcal{P}_\Delta, ct_{\mathsf{HD}}, \sigma_{\mathsf{HD}}, \ell_d)$**:** The verification algorithm computes $b \leftarrow \mathsf{BFR}.\mathbf{Ver}(vk, sk, \mathcal{P}_\Delta, ct_{\mathsf{HD}}, \sigma_{\mathsf{HD}}, \kappa)$ to verify the correctness of the outsourced computation. In our case of study, $\mathcal{P}_\Delta$ is a multi-labeled program [4] for the arithmetic circuit for calculating the encrypted HD. The $\mathsf{BFR}.\mathbf{Ver}$ algorithm essentially performs two integrity-checks:
$$ct_{\mathsf{HD}} = y_0 \mod (d + \ell_d) \tag{3}$$
$$W = e(g,g)^{y_0} \cdot e(Y_1, g)^{\theta} \cdot (\hat{Y}_2)^{\theta^2} \tag{4}$$
If the verification output is $b = 0$ the algorithm returns
$$(acc_{\mathsf{VC}}, acc_{\mathsf{HD}}) = (0, 0).$$
Otherwise, if $b = 1$, it proceeds with the biometric authentication check: it computes $w \leftarrow \mathsf{SHE.Dec(ct)}$ to retrieve the actual Hamming distance $w = \mathsf{HD}(A, B)$. If $\mathsf{HD}(A, B) \leq \kappa$, here κ corresponds to the accuracy of the BAP, the algorithm returns
$$(acc_{\mathsf{VC}}, acc_{\mathsf{HD}}) = (1, 1).$$
If $\mathsf{HD}(A, B) > \kappa$, the output is
$$(acc_{\mathsf{VC}}, acc_{\mathsf{HD}}) = (1, 0).$$

3.3 Correctness analysis

In our BFR+SHE scheme the outsourced function is the Hamming distance HD, that can be represented by a bi-variate deterministic quadratic function. Thus, we can avoid using gate-by-gate induction proofs, as done in [4], and demonstrate the correctness in a direct way. In what follows, we adopt the notation in [4], and we prove the correctness of BFR+SHE by walking through the arithmetic circuit of HD step by step.

Figure 3 depicts the arithmetic circuit for calculating the encrypted Hamming distance. A and B denote the encrypted stored and fresh biometric templates respectively. C_1 and C_2 are the constants in the function as defined in the $\mathsf{BFR{+}SHE}.\textbf{Comp}$ algorithm. The D letter indicates the -2 in the function, but since -2 is not in the valid range $\mathbb{Z}_p$ required by the original BFR scheme, we need to have an intermediate transformation of $D = d - 2$. All A, B, C_1 and C_2 are in $\mathbb{Z}_d$. Finally, the σs are the outcome tags of the form $\sigma^{(i)} = (y_0^{(i)}, Y_1^{(i)}, \hat{Y}_2^{(i)}) \in \mathbb{Z}_p \times \mathbb{G} \times \mathbb{G}_T$ after each gate operation, and the Rs are values in either $\mathbb{G}$ or $\mathbb{G}_\mathbb{T}$, which are used for homomorphic evaluation over bilinear groups (i.e., **GroupEval** in [4]). We let α and β be $\mathsf{vEnc}_1(A)$ and $\mathsf{vEnc}_2(B)$ and each of them has a tag: $\sigma_\alpha = (y_0^{(A)}, Y_1^{(A)}, 1)$ and $\sigma_\beta = (y_0^{(B)}, Y_1^{(B)}, 1)$. These two tags are generated by the $\mathsf{BFR}.\textbf{Auth}$ algorithm, which specifies that $y_0^{(A)} = \alpha$ and $Y_1^{(A)} = (R_\alpha \cdot g^{-\alpha})^{1/\theta}$. Similarly, we have $y_0^{(B)} = \beta$ and $Y_1^{(B)} = (R_\beta \cdot g^{-\beta})^{1/\theta}$. To verify the correctness of our $\mathsf{BFR{+}SHE}$ scheme, we need to check that the two equations specified in the $\mathsf{BFR}.\textbf{Ver}$ algorithm are satisfied if the computation is performed correctly. To this end, let $\sigma_{\mathsf{HD}} = (y_0^{(\mathsf{HD})}, Y_1^{(\mathsf{HD})}, \hat{Y}_2^{(\mathsf{HD})})$ be the final tag (which is equivalent to σ_6 in the arithmetic circuit depicted in Figure 3).

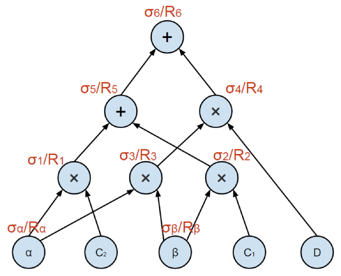

Fig. 3: The arithmetic circuit for calculating the encrypted Hamming distance.

The first step is to derive the tags for the intermediate calculation and eventually the final tag. If we run the $\mathsf{SHE}.\textbf{Eval}$ algorithm homomorphically through the circuit, we will get the outcome tags $\sigma_1, \ldots, \sigma_6$ (for details see Appendix A). We thus derive σ_{HD} (equivalent to σ_6):

$$\begin{aligned}
\sigma_{HD} &= (y_0^{(\mathsf{HD})}, Y_1^{(\mathsf{HD})}, \hat{Y}_2^{(\mathsf{HD})}) \\
&= (\quad C_2 \cdot y_0^{(A)} + C_1 \cdot y_0^{(B)} + D \cdot y_0^{(A)} \cdot y_0^{(B)}, \\
&\qquad\qquad (Y_1^{(A)})^{y_0^{(B)} \cdot D + C_2} \cdot (Y_1^{(B)})^{y_0^{(A)} \cdot D + C_1}, \quad e(Y_1^{(A)}, Y_1^{(B)})^D \quad).
\end{aligned}$$

Now we show the proofs for the two verification equations. First we need to prove Equation (3), i.e., $ct_{\mathsf{HD}} = y_0 \mod (d + \ell_d)$. The equality holds as for Equation (2). The end result is: $ct_{\mathsf{HD}} = y_0^{(\mathsf{HD})} \mod d + (\ell \mod d) \cdot (p \mod d)$. As we define $\ell_d = (\ell \mod d) \cdot (p \mod d)$, we can derive Equation (3). Secondly, we need to prove that Equation (4) holds, i.e., $W = e(R_\alpha^{C_2} \cdot R_\beta^{C_1}, g) \cdot e(R_\alpha, R_\beta)^D$. To this end, we run $\textbf{GroupEval}(f, R_\alpha, R_\beta)$ and execute the bilinear gate operations. Recall that R_α and R_β correspond to R_A and R_B in the notation used in the construction, Denote by R_6 the final result of running **GroupEval** over the circuit of HD. It holds that:

$$\begin{aligned} R_6 &= \mathbf{GroupEval}(f, R_\alpha, R_\beta) = e(R_\alpha^{C_2} \cdot R_\beta^{C_1}, g) \cdot e(R_\alpha, R_\beta)^D \\ &= \mathbf{GroupEval}(f, R_\alpha, R_\beta) = e(g,g)^{y_0^{\mathsf{HD}}} \cdot e(Y_1^{\mathsf{HD}}, g)^\theta \cdot (\hat{Y}_2^{(\mathsf{HD})})^{\theta^2} \end{aligned}$$

By expanding the last expression the desired result (see Appendix A for details). Thus, we have proved the correctness of the BFR+SHE scheme. which are the results of the pseudo-random function F_K in the BFR.**Ver** algorithm.

4 Improving the Yasuda et al. protocol

In this section, we describe a modified version of the Yasuda et al. [23] protocol that is secure against the recently identified *hill-climbing* attack that can be performed by a malicious computation server $\mathcal{CS}$. It is composed of four distributed parties: a client $\mathcal{C}$ (holding the keys pk, ek and vk), a computation server/database $\mathcal{CS}$ (holding the keys pk and ek), an authentication server $\mathcal{AS}$ (holding the keys pk, sk and vk). In the proposed protocol, we preserve the assumption that $\mathcal{AS}$ is a trusted party and furthermore assume the client $\mathcal{C}$ and the database $\mathcal{DB}$ are also trusted parties. $\mathcal{C}$ is responsible to manage the secret key vk for the verifiable computation scheme and $\mathcal{DB}$ stores the encrypted reference biometric templates with the identities of the corresponding clients. However, $\mathcal{CS}$ can be malicious and cheat with flawed computations. We describe the three main phases of our proposed improvement of Yasuda et al.'s privacy-preserving biometric authentication protocol:

Setup Phase: In this phase the authentication server $\mathcal{AS}$ runs SHE.**KeyGen**(λ) to generate the public key pk and the secret key sk of the somewhat homomorphic encryption (SHE) scheme. $\mathcal{AS}$ keeps sk and distributes pk to both the client $\mathcal{C}$ and the computation server $\mathcal{CS}$.

Enrollment Phase: Upon client registration, the client $\mathcal{C}$ runs BFR.**KeyGen**(λ) to generate the public evaluation key ek and the secret verification key vk. $\mathcal{C}$ distributes ek to $\mathcal{CS}$ and vk to $\mathcal{AS}$. The client $\mathcal{C}$ generates a 2048-bit feature vector A from the client's biometric data, runs BFR+SHE.**Enc**($pk, A, 0$) to obtain the ciphertext ct_A. $\mathcal{C}$ authenticates ct_A by running BFR+SHE.**Auth**(vk, L_A, ct_A) and outputs a tag σ_A. Then $\mathcal{C}$ sends the three-tuple $(\mathsf{ID}, ct_A, \sigma_A)$ to the database. This three-tuple serves as the reference biometric template for the specific client with identity ID.

Authentication Phase: The client provides fresh biometric data as a feature vector $B \in \{0,1\}^{2048}$. $\mathcal{C}$ runs $\mathsf{BFR{+}SHE}.\mathbf{Enc}(pk, B, 1)$ to obtain the ciphertext ct_B and authenticates it by running $\sigma_B \leftarrow \mathsf{BFR} + \mathsf{SHE}.\mathbf{Auth}(vk, L_B, ct_B)$. $\mathcal{C}$ sends $(\mathsf{ID}, ct_B, \sigma_B)$ to $\mathcal{CS}$, who extracts the tuple $(\mathsf{ID}, ct_A, \sigma_A)$ corresponding to the client to be authenticated (using the ID as the search key). $\mathcal{CS}$ calculates the encrypted Hamming distance $ct_{\mathsf{HD}} \leftarrow \mathsf{BFR} + \mathsf{SHE}.\mathbf{Comp}(pk, ct_A, ct_B)$ and generates a corresponding tag $\sigma_{\mathsf{HD}} \leftarrow \mathsf{BFR} + \mathsf{SHE}.\mathbf{Eval}(ek, pk, \sigma_A, \sigma_B)$. Then, $\mathcal{CS}$ sends $(\mathsf{ID}, ct_{\mathsf{HD}}, \sigma_{\mathsf{HD}})$ to the authentication server. $\mathcal{AS}$ runs $(acc_{\mathsf{VC}}, acc_{\mathsf{HD}}) \leftarrow \mathsf{BFR{+}SHE}.\mathbf{Ver}(vk, sk, \mathcal{P}_\Delta, ct_{\mathsf{HD}}, \sigma_{\mathsf{HD}}, \kappa)$, where κ is the desired accuracy level of the BAP. If either acc_{VC} or acc_{HD} is 0 $\mathcal{AS}$ outputs a no, for authentication rejection. Otherwise, $(acc_{\mathsf{VC}}, acc_{\mathsf{HD}}) = (1,1)$ and $\mathcal{AS}$ outputs yes, for authentication success.

4.1 Security analysis of the proposed BAP

Our primary aim is to demonstrate that our privacy-preserving biometric authentication protocol is not vulnerable to Abidin et al.'s template recovery attack [1]. To this end, we sketch the attack setting in Figure 4.

Input: the client's identity ID, the public key for SHE scheme pk, the public evaluation key for the BFR scheme ek, the stored reference template and tag $(\mathsf{vEnc}_1(A), \sigma_A)$, the encrypted fresh template $\mathsf{vEnc}_2(B)$ and its tag σ_B.

Output: the inner product $ct_P = \mathsf{vEnc}_1(A) \cdot \mathsf{vEnc}_2(A')$ and the tag σ'_{HD}.

Goals: make the authentication server accept the inner product computation and return yes, and use the hill-climbing strategy recover the reference template A.

Fig. 4: Setting for Abidin et al.'s template recovery attack in [1].

We recall that for this attack, the adversary is a malicious computational server who tries to recover the stored reference biometric template of a client with identity ID. All the other parties of the BAP, are trusted and behave honestly.

In what follows, we show that the malicious $\mathcal{CS}$ cannot forge a tag $\sigma_{\mathsf{HD}'}$ that passes the verification checks performed in BFR. It is possible for the adversary to cheat on the first equality (Equation (3)) as it only tests that the returned computation result (ct_{HD} or ct_P) aligns with the arithmetic circuit used to generate the tag (σ_{HD} or $\sigma_{\mathsf{HD}'}$). In [1], $\mathcal{CS}$ succeeds by computing the arithmetic circuit for the inner product instead of HD.

In this case, it is not possible for the malicious computational server to fool the second second integrity check (Equation (4)). In details, $\mathcal{AS}$ calculates $W = \mathbf{GroupEval}(f, R_\alpha, R_\beta)$, and since $\mathcal{AS}$ is honest, $f = \mathsf{HD}$ is the arithmetic circuit for the Hamming distance. If $\mathcal{CS}$ returns incorrect results, with overwhelming probability the second verification equation does not hold, thus the attack is mitigated.

Other threats. In what follows, we consider attack scenarios in which one of the participating entities in the BAP is malicious.

Malicious client. $\mathcal{C}$ is responsible to capture the reference template and the fresh template as well as to perform the encryption. If the client is malicious, the knowledge of the encryption secret key and of the identity ID enables $\mathcal{C}$ to initiate a *center search attack* and recover the stored template A as explained in [17]. Unfortunately, Pagnin et al. [17] show that this class of attacks cannot be detected using verifiable computation techniques, since the attacker is not cheating with the computation.

A new concern with the modified Yasuda et al. protocol is the key generation for BFR. In the protocol, we let the client $\mathcal{C}$ generate the private key vk, the evaluation key ek and the authentication tags because we assume $\mathcal{C}$ is a trusted party. If $\mathcal{C}$ turns malicious, it could give a fake vk to the authentication server $\mathcal{AS}$ and initiate the template recovery attack with the inner product by simulating $\mathcal{CS}$. Since the adversary ($\mathcal{C}$) controls vk, the computation verification step becomes meaningless.

Malicious computation server. The main motivation to integrate VC in BAPs is indeed to prevent $\mathcal{CS}$ from behaving dishonestly. Unlike the client $\mathcal{C}$, $\mathcal{CS}$ only has access to the encrypted templates $\mathsf{vEnc}_1(A)$ and $\mathsf{vEnc}_2(B)$ and the user pseudonyms. $\mathcal{CS}$ cannot modify the secret key of the BFR scheme. We have analysed how the template recovery attack conducted by $\mathcal{CS}$ can be countered and hence we shorten the discussion here.

In contrast to the original protocol, $\mathcal{CS}$ needs to calculate an extra value ℓ_d to solve the range issue after integrating BFR. However, ℓ_d is still operated on the ciphertext level and is not involved in the second verification equation. Thus learning ℓ_d does not provide any significant advantage in recovering the templates.

Malicious authentication server. A malicious $\mathcal{AS}$ will completely break down the privacy of the BAP since it controls the secret key sk used by the SHE scheme. If $\mathcal{AS}$ successfully eavesdrops and obtains the ciphertext $\mathsf{vEnc}_1(A)$ or $\mathsf{vEnc}_2(B)$, it can recover the plaintext biometric templates.

4.2 Efficiency Analysis

The original BFR scheme in [4] allows alternative algorithms to improve the efficiency of the verifier. Although in our instantiation we did not use these algorithms, the current definition of the multi-labels in BFR+SHE is extensible. Given also that the function to be computed is $f = \mathsf{HD}$ and has a very simple description as arithmetic circuit, running the BFR+SHE.**Ver** algorithm requires $O(|f|)$ computational time. In addition, if the amortized closed-form efficiency functionality is adopted, the verification function will run in time $O(1)$. Nonetheless, the arithmetic circuit of HD has 6 gates only and the saved computation overhead would be relatively small.

5 A flawed approach

Privacy and integrity are the two significant properties desired in a privacy-preserving BAP. There are two possible ways to combine VC and homomorphic encryption (HE): running VC on top of HE, and viceversa, running HE on top of VC.

In the first case, the data (biometric template) is first encrypted and then encoded to generate an authentication proof. Our construction of BFR+SHE follows this principle. In this approach, $\mathcal{AS}$ can make the judgement whether the output of $\mathcal{CS}$ is from a correct computation of HD *before* decrypting the ciphertext.

In the second case, the data is first encoded for verifiable computation and then the encoded data is encrypted. This combination is not really straightforward and is prune to security breaches.

In this section, we demonstrate an attack strategy that may lead to information leakage in case the homomorphic encryption scheme (henceforth FHE)[1] is applied on top of a VC scheme. For the sake of generality, we define $\mathsf{FHE} = (KeyGen_{FHE}, Enc, Dec, Eval)$. For verifiable computation scheme we adopt the notation of Gennaro et al. [11] and define $\mathsf{VC} = (KeyGen_{VC}, ProbGen, Compute, Ver)$, where $KeyGen_{VC}$ outputs the private key sk_{vc} and public key pk_{vc}; $ProbGen$ takes sk_{vc} and the plaintext x as input and outputs the encoded value σ_x; $Compute$ takes the circuit f, the encrypted encoded input and outputs the encoded version of the output; Ver is performed to verify the correctness of the computation given the secret key sk_{vc} and the encoded output σ_y. The main idea of the flawed approach is to first encode the data in plaintext and then encrypt the encoded data. It can be represented by $\hat{x} = Enc(ProbGen(x))$, where $\hat{x}$ is what the malicious server gets access to.

[1] The same leakage of information could happen if a SHE scheme is used.

5.1 The attack

We describe now a successful attack strategy to break the privacy-preservation property of a BAP built with the second composition method: HE on top of VC (or HE *after* VC). The adversary's goal is to recover σ_y, i.e., the encoded value of the computation result. The attack runs in different phases. We show that the privacy-preserving property is broken if $q \geq n$, where q is the number of queries in the learning phase and n is the length of the encoded result σ_y. For simplicity we collect the two entities $\mathcal{C}$ and $\mathcal{AS}$ into a single trusted party $\mathcal{V}$ that we refer to as the Verifier.

The attack is depicted in Figure 5, a more detailed description follows.

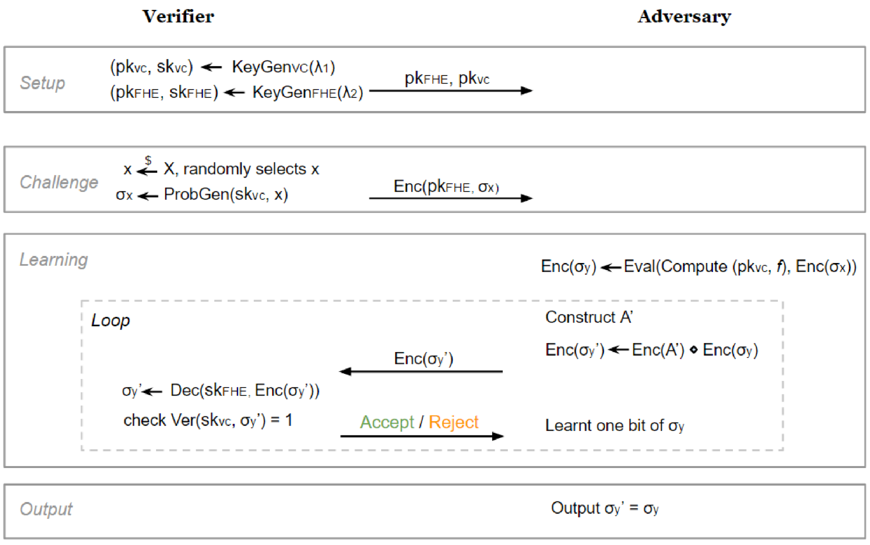

Fig. 5: An attack strategy against the naïve the integration of FHE on top of VC.

Setup phase: $\mathcal{V}$ generates the keys of the protocol and gives pk_{vc}, pk_{FHE} to $\mathcal{A}$.

Challenge phase: $\mathcal{V}$ generates the encoded version σ_x for the input x. $\mathcal{V}$ encrypts the encoded input and sends $Enc(\sigma_x)$ to $\mathcal{A}$.

Learning phase: $\mathcal{A}$ uses $\mathcal{V}$ as a decryption oracle by sending verification queries, which can be further divided into the following steps:

1. $\mathcal{A}$ performs honest computations and derives the $Enc(\sigma_y)$.
2. $\mathcal{A}$ constructs a vector $A' \in \mathbb{Z}_2^n$ equal in length to σ_y. A' is initialized with the last bit set to 0 and the rest of the bits set to 1. For the i^{th} trial, we set $A' = (1_1, \ldots, 0_i, 1_{i+1}, \ldots, 1_{n-1}, 1_n)$, i.e., set the i^{th} bit to 0 and the rest bits to 1.

3. $\mathcal{A}$ encrypts the tailored vector A' and reuses the honest result $Enc(\sigma_y)$ from step 1. Then she computes: $Enc(\sigma_{y'}) = Enc(A') \diamond Enc(\sigma_y)$, where $\diamond$ represents the Hadamard product for binary vectors and sends the result to $\mathcal{V}$ for verification.
4. $\mathcal{V}$ decrypts $Enc(\sigma'_y)$. Thanks to the homomorphic property of FHE, $\mathcal{V}$ can derive $\sigma_{y'} = A' \diamond \sigma_y$. $\mathcal{V}$ checks the computation based on the encoded result $\sigma_{y'}$ and returns either *accept* if $Ver(sk, \sigma_{y'}) = 1$ or *reject*, otherwise.
5. The attacker A' acts as a "mask": it copies all the bit values of σ_y into $\sigma_{y'}$ except for the i^{th} bit, which is always set to zero. Consequently, if the output of the verification is *accept*, $\mathcal{A}$ will learn that $\sigma_y = \sigma_{y'}$ as well as $Enc(\sigma_y) = Enc(\sigma_{y'})$, which reveals that the i^{th} bit of σ_y equals to 0. Similarly, if the output of the verification is *reject*, $\mathcal{A}$ learns that the i^{th} bit of σ_y is 1. In both cases, one bit of σ_y is leaked.

Output phase: After $q \geq n$ verification queries, where n equals the length of σ_y, $\mathcal{A}$ outputs $\sigma_{y'}$.

It is trivial to check that that $\sigma_{y'} = \sigma_y$ and thus $Ver(sk, \sigma'_y) = Ver(sk, \sigma'_y) = 1$ and attacker's goal is achieved.

The attack demonstrates that the order of combining a VC and a (F)HE is very crucial: the verifier must decrypt the ciphertext *before* it can determine whether it is the result of the correct outsourced computation. Adopting such a scheme in a BAP would make $\mathcal{AS}$ a decryption oracle. Leaking information on the Hamming distance may be exploited to perform further attacks that might lead to the full recovery of biometric templates as it has been recently shown [17]. Formally speaking, we can say that the HE on top of VC is not a chosen-ciphertext attack (CCA) secure scheme.

6 Conclusions

Biometric authentication protocols have gained considerable popularity for access control services. Preserving the privacy of the biometric templates is highly critical due to their irrevocable nature. Yasuda et al. proposed a biometric authentication protocol [23] using a SHE scheme. However, a hill-climbing attack [1] has been presented against this protocol that relies on a malicious internal computation server $\mathcal{CS}$ that performs erroneous computations and leads to the disclosure of the biometric reference template. We counter the aforementioned attack by constructing a new scheme named BFR+SHE which adds a verifiable computation layer to the SHE scheme. We then describe a modified version of the Yasuda et al. protocol that utilizes our BFR+SHE scheme, and demonstrate that the improved BAP provides higher privacy guarantees. Although employing VC to mitigate hill-climbing attack techniques seems a quite straightforward step, we demonstrate that not all combinations of a VC scheme with a HE one are secure, and show how a naïve combination leads to a drastic private information leakage in BAP.

7 Acknowledgements

This work was partially supported by the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement no 608743; the VR grant PRECIS no 621-2014-4845 and the STINT grant "Secure, Private & Efficient Healthcare with wearable computing" no IB2015-6001.

References

1. A. Abidin and A. Mitrokotsa. Security aspects of privacy-preserving biometric authentication based on ideal lattices and ring-lwe. In *Proceedings of the IEEE Workshop on Information Forensics and Security 2014 (WIFS 2014)*, 2014.
2. L. Babai. Trading group theory for randomness. In *Proceedings of STOC'85*, pages 421–429, New York, NY, USA, 1985. ACM.
3. M. Backes, M. Barbosa, D. Fiore, and R. M. Reischuk. Adsnark: Nearly practical and privacy-preserving proofs on authenticated data. In *Proceedings of the 36th IEEE Symposium on Security and Privacy (Oakland)*, 2015.
4. M. Backes, D. Fiore, and R. M. Reischuk. Verifiable delegation of computation on outsourced data. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 863–874. ACM, 2013.
5. M. Barbosa, T. Brouard, S. Cauchie, and S. M. de Sousa. Secure biometric authentication with improved accuracy. In *ACISP*, volume 5107 of *LNCS*, pages 21–36. Springer, 2008.
6. J. Bringer, H. Chabanne, F. Kraïem, R. Lescuyer, and E. Soria-Vázquez. Some applications of verifiable computation to biometric verification. In *Information Forensics and Security (WIFS), 2015 IEEE International Workshop on*, pages 1–6. IEEE, 2015.
7. J. Bringer, H. Chabanne, and A. Patey. Privacy-preserving biometric identification using secure multiparty computation: An overview and recent trends. *Signal Processing Magazine, IEEE*, 30(2):42–52, 2013.
8. J. Bringer, H. Chabanne, and A. Patey. Shade: Secure hamming distance computation from oblivious transfer. In *Financial Cryptography and Data Security*, pages 164–176. Springer, 2013.
9. C. Costello, C. Fournet, J. Howell, M. Kohlweiss, B. Kreuter, M. Naehrig, B. Parno, and S. Zahur. Geppetto: Versatile verifiable computation. In *2015 IEEE Symposium on Security and Privacy*, pages 253–270. IEEE, 2015.

10. D. Fiore, R. Gennaro, and V. Pastro. Efficiently verifiable computation on encrypted data. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 844–855. ACM, 2014.
11. R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Advances in Cryptology–CRYPTO 2010*, pages 465–482. Springer, 2010.
12. C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.
13. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, Feb. 1989.
14. V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In *Cryptology and Network Security*, pages 1–20. Springer, 2009.
15. M. Naehrig, K. Lauter, and V. Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 113–124. ACM, 2011.
16. E. Pagnin. *Authentication under Constraints*. Licentiate dissertation, Chalmers University of Technology, 2016.
17. E. Pagnin, C. Dimitrakakis, A. Abidin, and A. Mitrokotsa. On the leakage of information in biometric authentication. In *Progress in Cryptology–INDOCRYPT 2014*, pages 265–280. Springer, 2014.
18. E. Pagnin and A. Mitrokotsa. Privacy-preserving biometric authentication: challenges and directions. *IACR Cryptology ePrint Archive*, 2017:450, 2017.
19. B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly practical verifiable computation. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, pages 238–252, Washington, DC, USA, 2013. IEEE Computer Society.
20. K. Simoens, J. Bringer, H. Chabanne, and S. Seys. A framework for analyzing template security and privacy in biometric authentication systems. *Information Forensics and Security, IEEE Transactions on*, 7(2):833–841, 2012.
21. K. Simoens *et al.* A framework for analyzing template security and privacy in biometric authentication systems. *IEEE Transactions on Information Forensics and Security*, 7(2):833–841, 2012.
22. A. Stoianov. Cryptographically secure biometrics. *SPIE 7667, Biometric Technology for Human Identification VII*, pages 76670C–12, 2010.
23. M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama, and T. Koshiba. Practical packing method in somewhat homomorphic encryption. In *Data Privacy Management and Autonomous Spontaneous Security*, pages 34–50. Springer, 2014.

A Details in the correctness analysis

In this section, we show the intermediate steps of the calculation.
The derived tags are:

$$\sigma_1 = (C_2 \cdot y_0^{(A)}, (Y_1^{(A)})^{C_2}, 1);\ \sigma_2 = (C_1 \cdot y_0^{(B)}, (Y_1^{(B)})^{C_1}, 1);$$
$$\sigma_3 = (y_0^{(A)} \cdot y_0^{(B)}, (Y_1^{(A)})^{y_0^{(B)}} \cdot (Y_1^{(B)})^{y_0^{(A)}}, e(Y_1^{(A)}, Y_1^{(B)}));$$
$$\sigma_4 = (D \cdot y_0^{(A)} \cdot y_0^{(B)}, (Y_1^{(A)})^{y_0^{(B)} \cdot D} \cdot (Y_1^{(B)})^{y_0^{(A)} \cdot D}, e(Y_1^{(A)}, Y_1^{(B)}))^D;$$
$$\sigma_5 = (C_2 \cdot y_0^{(A)} + C_1 \cdot y_0^{(B)}, (Y_1^{(A)})^{C_2} \cdot (Y_1^{(B)})^{C_1}, 1);$$

$$\sigma_6 = \begin{pmatrix} C_2 \cdot y_0^{(A)} + C_1 \cdot y_0^{(B)} + D \cdot y_0^{(A)} \cdot y_0^{(B)} \\ (Y_1^{(A)})^{y_0^{(B)} \cdot D + C_2} \cdot (Y_1^{(B)})^{y_0^{(A)} \cdot D + C_1} \\ e(Y_1^{(A)}, Y_1^{(B)})^D \end{pmatrix}$$

The homomorphic bilinear map calculation results are:

$$R_1 = R_\alpha^{C_2};\ R_2 = R_\beta^{C_1};\ R_3 = e(R_\alpha, R_\beta);\ R_4 = e(R_\alpha, R_\beta)^D;$$
$$R_5 = R_\alpha^{C_2} \cdot R_\beta^{C_1};\ R_6 = e(R_\alpha^{C_2} \cdot R_\beta^{C_1}, g) \cdot e(R_\alpha, R_\beta)^D.$$

To prove that $W = \mathbf{GroupEval}(f, R_\alpha, R_\beta)$ satisfies Equation (4), we start by analysing the three factors that made up the righthand of the equation, namely: $e(g,g)^{y_0^{\mathsf{HD}}} \cdot e(Y_1^{\mathsf{HD}}, g)^\theta \cdot (\hat{Y}_2^{(\mathsf{HD})})^{\theta^2}$. We in turn expand each one of the factors and finally compute the product of the results, evaluating it against W.
The first factor can be expanded as:

$$e(g,g)^{y_0^{HD}} = e(g,g)^{C_2 \cdot y_0^{(A)} + C_1 \cdot y_0^{(B)} + D \cdot y_0^{(A)} \cdot y_0^{(B)}} = e(g,g)^{C_2\alpha + C_1\beta + \alpha\beta D}.$$

The second factor is expanded as:

$$\begin{aligned} e(Y_1^{HD}, g)^\theta &= e((Y_1^{(A)})^{y_0^{(B)} \cdot D + C_2} \cdot (Y_1^{(B)})^{y_0^{(A)} \cdot D + C_1}, g)^\theta \\ &= e((R_\alpha \cdot g^{-\alpha})^{(\beta D + C_2)/\theta} \cdot (R_\beta \cdot g^{-\beta})^{(\alpha D + C_1)/\theta}, g)^\theta \\ &= e(R_\alpha^{\beta D + C_2} \cdot R_\beta^{\alpha D + C_1} \cdot g^{-2\alpha\beta D - \alpha C_2 - \beta C_1}, g) \\ &= e(R_\alpha, g)^{\beta D + C_2} \cdot e(R_\beta, g)^{\alpha D + C_1} \cdot e(g,g)^{-2\alpha\beta D - \alpha C_2 - \beta C_1}. \end{aligned}$$

The third factor is expanded as:

$$\begin{aligned} (\hat{Y}_2^{(\mathsf{HD})})^{\theta^2} &= e(Y_1^{(A)}, Y_1^{(B)})^{D\theta^2} = e((R_\alpha \cdot g^{-\alpha})^{1/\theta}, (R_\beta \cdot g^{-\beta})^{1/\theta})^{D\theta^2} \\ &= e(R_\alpha \cdot g^{-\alpha}, R_\beta \cdot g^{-\beta})^D = e(R_\alpha, R_\beta \cdot g^{-\beta})^D \cdot e(g^{-\alpha}, R_\beta \cdot g^{-\beta})^D \\ &= e(R_\alpha, R_\beta)^D \cdot e(R_\alpha, g)^{-\beta D} \cdot e(R_B, g)^{-\alpha D} \cdot e(g,g)^{\alpha\beta D}. \end{aligned}$$

Here we need to prove the right hand side is equal to W. We use a temporary variable $P = e(g,g)^{y_0^{\mathsf{HD}}} \cdot e(Y_1^{\mathsf{HD}}, g)^\theta \cdot (\hat{Y}_2^{(\mathsf{HD})})^{\theta^2}$ to denote the expansion result of the righthand-side. The expression below proves the correctness of the second verification equation (4).

$$\begin{aligned} P = {} & e(g,g)^{C_2 \cdot \alpha + C_1 \cdot \beta + D \cdot \alpha \cdot \beta} \cdot e(R_\alpha, g)^{\beta D + C_2} \cdot e(R_\beta, g)^{\alpha D + C_1} \cdot e(g,g)^{-2\alpha\beta D - \alpha C_2 - \beta C_1} \cdot \\ & \cdot e(R_\alpha, R_\beta)^D \cdot e(R_\alpha, g)^{-\beta D} \cdot e(R_B, g)^{-\alpha D} \cdot e(g,g)^{\alpha\beta D} \\ = {} & e(g,g)^0 \cdot e(R_\alpha, g)^{C_2} \cdot e(R_\beta, g)^{C_1} \cdot e(R_a, R_b)^D \\ = {} & e(R_\alpha^{C_2}, g) \cdot e(R_\beta^{C_1}, g) \cdot e(R_a, R_b)^D \\ = {} & e(R_\alpha^{C_2} \cdot R_\beta^{C_1}, g) \cdot e(R_a, R_b)^D = W. \end{aligned}$$